ʕ •ᴥ• ʔ

b34r5hell

Cryptography

Agenda

- What is Cryptography?
- Text Encodings: Hexadecimal and Base64
- Cyberchef
- Classical Ciphers
- Mathematical Ciphers
- RSA

ʕ•ᴥ•ʔ

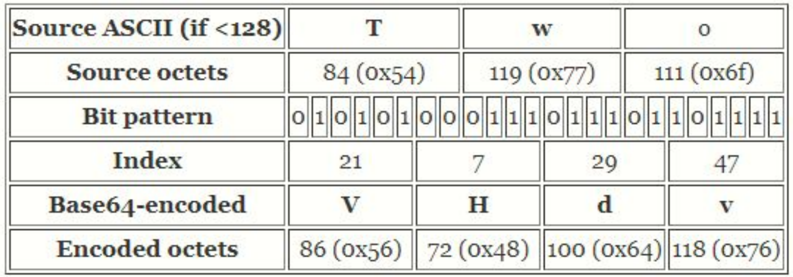
What is Cryptography?

- **Study and practice of secure communication techniques**
  - Prevent unauthorized people from viewing private messages
  - Can also verify the author and authenticity of a message
  - Uses various protocols (sometimes very math-heavy!) depending on the application
  - Plaintext is encrypted to become Ciphertext
  - Ciphertext is decrypted to become Plaintext

- **Example Situations**
  - Using Base64 or Hex to simplify a string of text
  - Making a secure connection to a website (HTTPS)
  - Keeping your passwords safe in the event of a data breach
  - Cryptocurrencies

Text Encodings: Hex and Base64

- Typical English characters are ascii and are stored as one byte (8 bits)
- **Hexadecimal** is a base-16 representation of data, as compared to binary (base-2) or decimal (base-10)
  - Uses character set 0-9 and a-f (16 characters)
  - Convenient because 16 is a power of 2, so every two characters represents one byte
- **Base64** is a text-encoding of data
  - Base64 encodes 6 bits with one character
  - Uses 64 character set (2^6 = 64)
  - Padding (# of bits is not a factor of 6)
  - Ex: “0xdeadbeef” (hex) -> “776t3g==” (Base64)

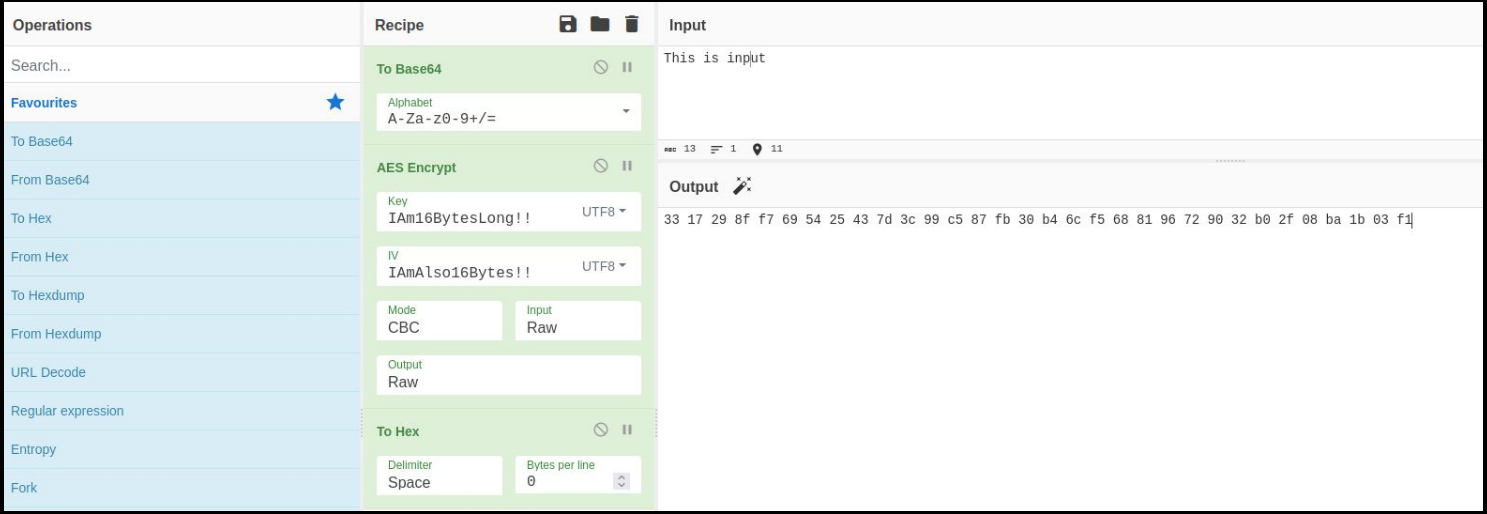


Source ASCII (if <128)	**T**								**w**								**o**							
Source octets	84 (0x54)								119 (0x77)								111 (0x6f)							
Bit pattern	0	1	0	1	0	1	0	0	0	1	1	1	0	1	1	1	0	1	1	0	1	1	1	1
Index	21						7						29						47					
Base64-encoded	**V**						**H**						**d**						**v**					
Encoded octets	86 (0x56)						72 (0x48)						100 (0x64)						118 (0x76)					

ʕ•ᴥ•ʔ

Cyberchef (https://gchq.github.io/CyberChef/)

- “CTF Swiss-Army-Knife”
  - Useful for all types of string manipulation and encryption
  - Can chain multiple manipulations together using GUI
  - Can be an alternative to Python scripting



To Base64
Alphabet: A-Za-z0-9+/=

AES Encrypt
Key: IAm16BytesLong!! (UTF8)
IV: IAmAlso16Bytes!! (UTF8)
Mode: CBC
Input: Raw
Output: Raw

To Hex
Delimiter: Space
Bytes per line: 0

Input

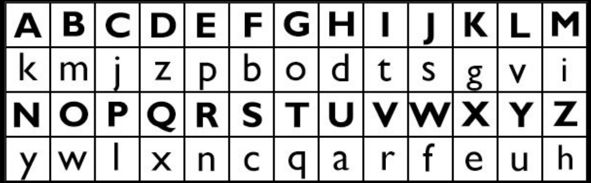
```
This is input
```

Output

```
33 17 29 8f f7 69 54 25 43 7d 3c 99 c5 87 fb 30 b4 6c f5 68 81 96 72 90 32 b0 2f 08 ba 1b 03 f1
```

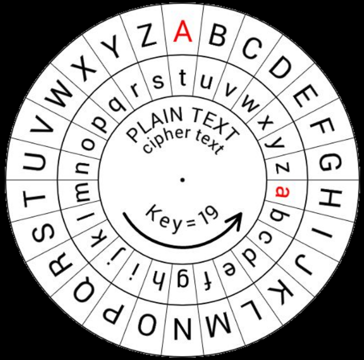
Classical Ciphers

- Substitution Cipher
  - Applied to a certain alphabet
  - Each letter is “substituted” with another unique letter
  - Subject to frequency analysis attacks



A	B	C	D	E	F	G	H	I	J	K	L	M
k	m	j	z	p	b	o	d	t	s	g	v	i
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
y	w	l	x	n	c	q	a	r	f	e	u	h

- Caesar Cipher
  - Technically a substitution cipher
  - Each letter is rotated 3 letters in the alphabet
  - More generic cipher: ROT (Caesar cipher ROT3)
  - ROT13 - encryption and decryption are the same


A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
t u v w x y z a b c d e f g h i j k l m n o p q r s
PLAIN TEXT
cipher text
Key=19

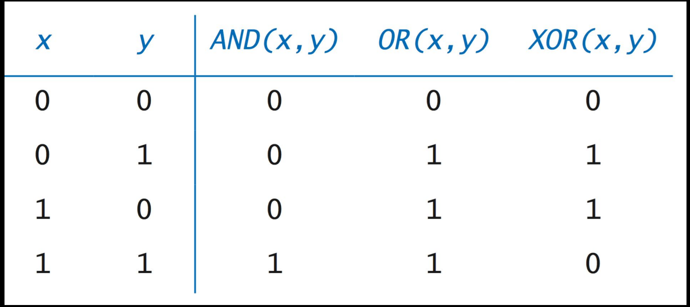
- Many other classical ciphers
  - All have an agreed mutation to the ciphertext that can be reversed with easily bruteforcable (normally) information

XOR

- One of the fundamental binary “bitwise” operations
  - Others include OR, AND

- Given two binary numbers, pair up each bit
  - If they are the same, output 0
  - If they are different, output 1

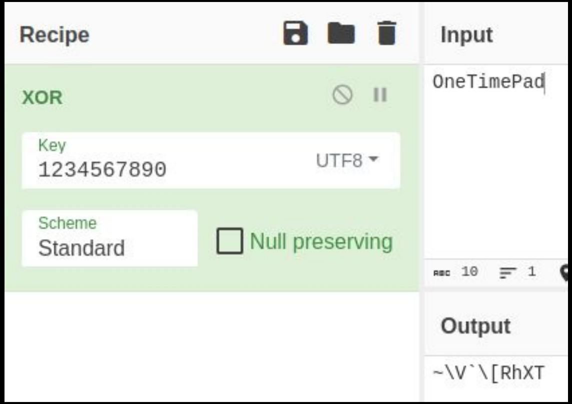
- Performing XOR twice undoes it
  - **10101** ^ 11000 = 01101
  - 01101 ^ 11000 = **10101**



x	*y*	*AND(x,y)*	*OR(x,y)*	*XOR(x,y)*
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

One-Time-Pad (OTP) Encryption

- **Single-use pre-shared key**
- **XOR the plaintext with the key to encrypt**
  - Each bit has 50% chance to be 1, 50% chance to be 0
  - Not the case with things like AND, OR
- **XOR the ciphertext with the key to decrypt**
  - Takes advantage of XOR being its own inverse
- **“Perfectly Secure” if:**
  - Key is “truly random” and kept completely secret
  - Key is at least as long as plaintext
  - Key is not reused (reused keys can be mathematically exploited)


Recipe

XOR

Key: 1234567890 (UTF8)

Scheme: Standard

□ Null preserving

Input

```
OneTimePad
```

Output

```
~\V`\[RhXT
```

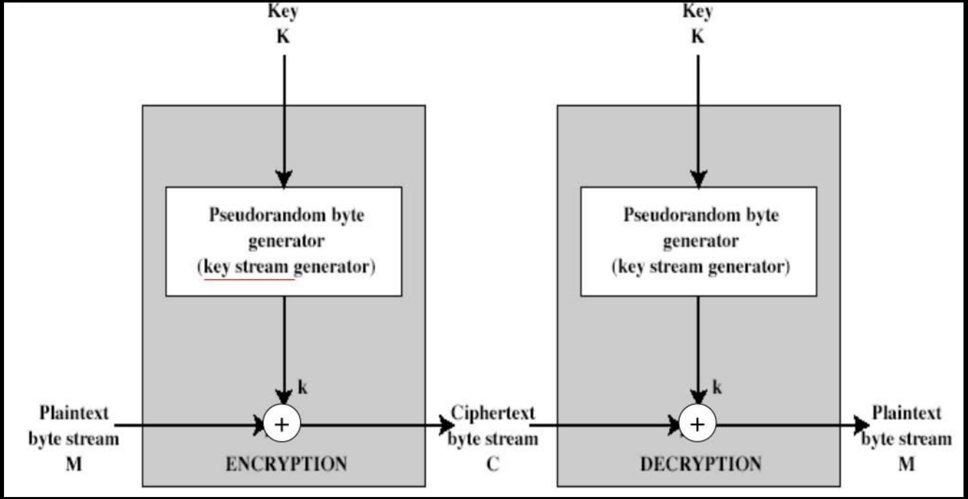
Stream Cipher

- **Inspired by One-Time Pad** (Uses bitwise XOR)

- Aim to remove the “key-length >= plaintext length” restriction

- Initial small key, key is “stretched” using various PRNG schemes

- Not “perfectly secure” like OTP, but more practical



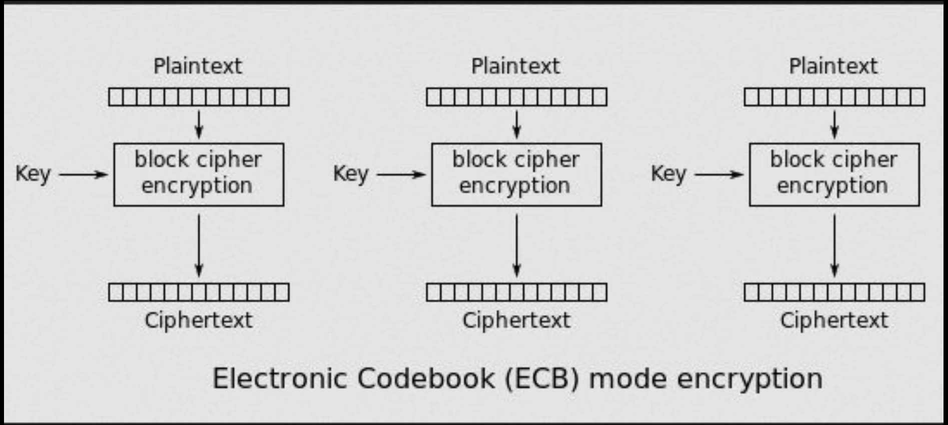
Block Cipher

- Split plaintext into "blocks" of bits. Encrypt each block with key to form ciphertext.

- Block size can vary depending on encryption scheme

- Encryption Schemes:
  - DES
  - AES (common)

Block 1 Block 2 Block 3

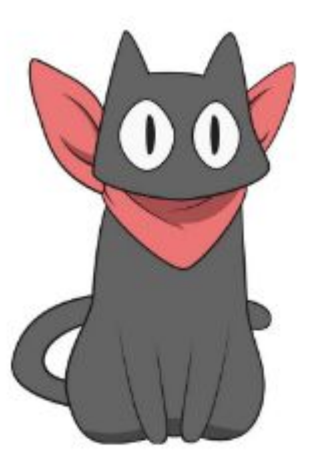


Electronic Codebook (ECB) mode encryption

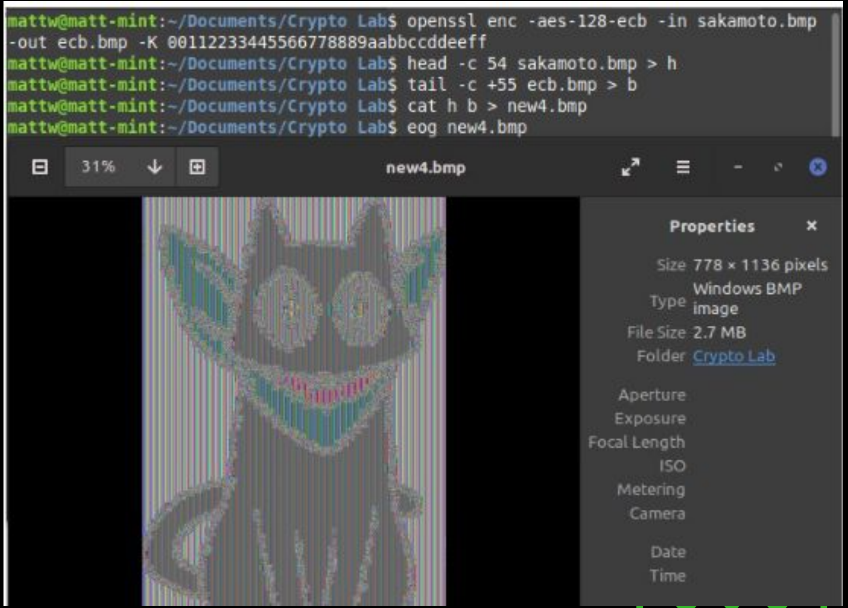
("block cipher encryption" in image)

Issue with Naive Block Ciphers

- What if plaintext blocks are the same?
- If using the same input to encryption, ciphertext blocks will also be the same!
- Noticeable on uncompressed images (PNG, BMP)
- Need a way to somehow differentiate plaintext!



sakamoto.bmp



```
mattw@matt-mint:~/Documents/Crypto Lab$ openssl enc -aes-128-ecb -in sakamoto.bmp
-out ecb.bmp -K 00112233445566778889aabbccddeeff
mattw@matt-mint:~/Documents/Crypto Lab$ head -c 54 sakamoto.bmp > h
mattw@matt-mint:~/Documents/Crypto Lab$ tail -c +55 ecb.bmp > b
mattw@matt-mint:~/Documents/Crypto Lab$ cat h b > new4.bmp
mattw@matt-mint:~/Documents/Crypto Lab$ eog new4.bmp
```

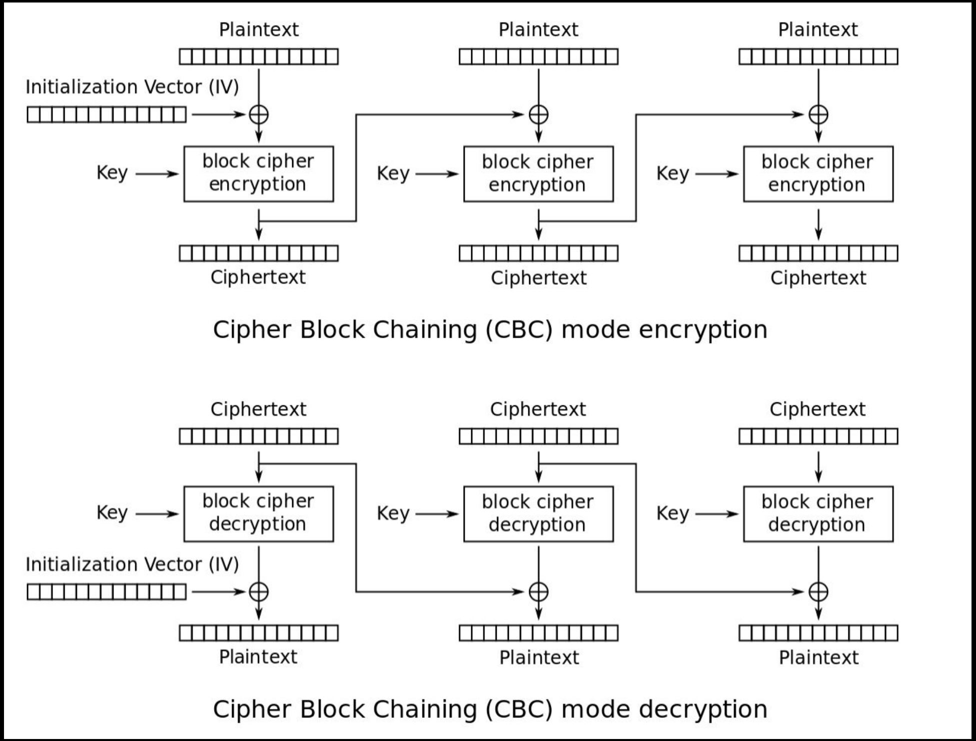
Size 778 × 1136 pixels
Type Windows BMP image
File Size 2.7 MB
Folder Crypto Lab
Aperture
Exposure
Focal Length
ISO
Metering
Camera
Date
Time

Making Block Cipher Secure

- **“Chain” blocks together**
  - Requires an initialization vector (IV) to start the chaining

- XOR one block’s ciphertext with next block’s plaintext before encryption
  - Opposite for decryption

- Even if plaintext has identical blocks, ciphertext blocks will differ greatly
  - Avalanche Effect - small change in input leads to large change in output



Cipher Block Chaining (CBC) mode encryption

Cipher Block Chaining (CBC) mode decryption

openssl

- Command line tool to perform lots of cryptographic functions
- Using AES
    - Ex:
        - `openssl enc -e -aes-128-ebc -in input.txt -out output.bin -K 00112233445566778899aabbccddeeff`
    - AES always has a block size of 128 bits (16 bytes)
    - AES can have varying key sizes (128, 192, and 256 bits)
    - Different chaining methods can be specified

RSA

- Common example of asymmetric encryption
    - Encrypt with the public key, decrypt with the private key
    - Can also sign things - prove you own the private key corresponding with a public key
        - Sign with private key, check signature with public key
- Relies on number theory and finite fields formed by large prime numbers
- One of the many areas of crypto to explore further!

Tasks

- Download Ghidra or your preferred decompiler (for next meeting)
- Complete the associated dojo module